

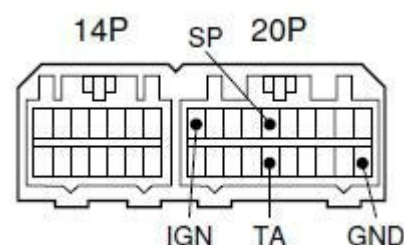
タコメータV2(tachometer)

概要

以前にも、タコメータ(tachometer)を製作しました。この時には、シガーライターソケット(cigarette lighter socket)から、電源を取り、その電源に含まれるオルタネータの微小信号波形(三相交流)より、エンジンの回転数を求めました。

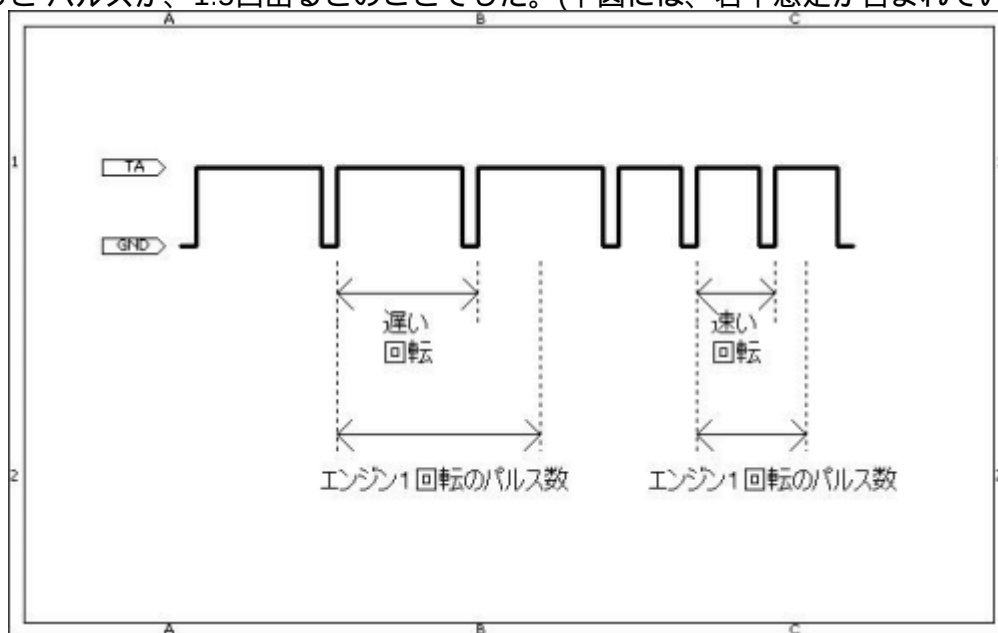
しかし、最近の車に搭載されている、エンジンコンピュータ(ECU)からはTA信号(エンジン回転数)が、標準で出力されているようなので、これを使ったタコメータを製作してみました。

動作原理



<ECUからの信号線の例> ホンダのトゥディ(排気量(660)、型式JA4)

<TA信号の波形の例> ホンダのトゥディ(排気量(660)、型式JA4)メーカーに問い合わせた限りでは、エンジンが1回転するとパルスが、1.5回出るとのことでした。(下図には、若干想定が含まれています)



のでご了承ください)

<処理の流れ>

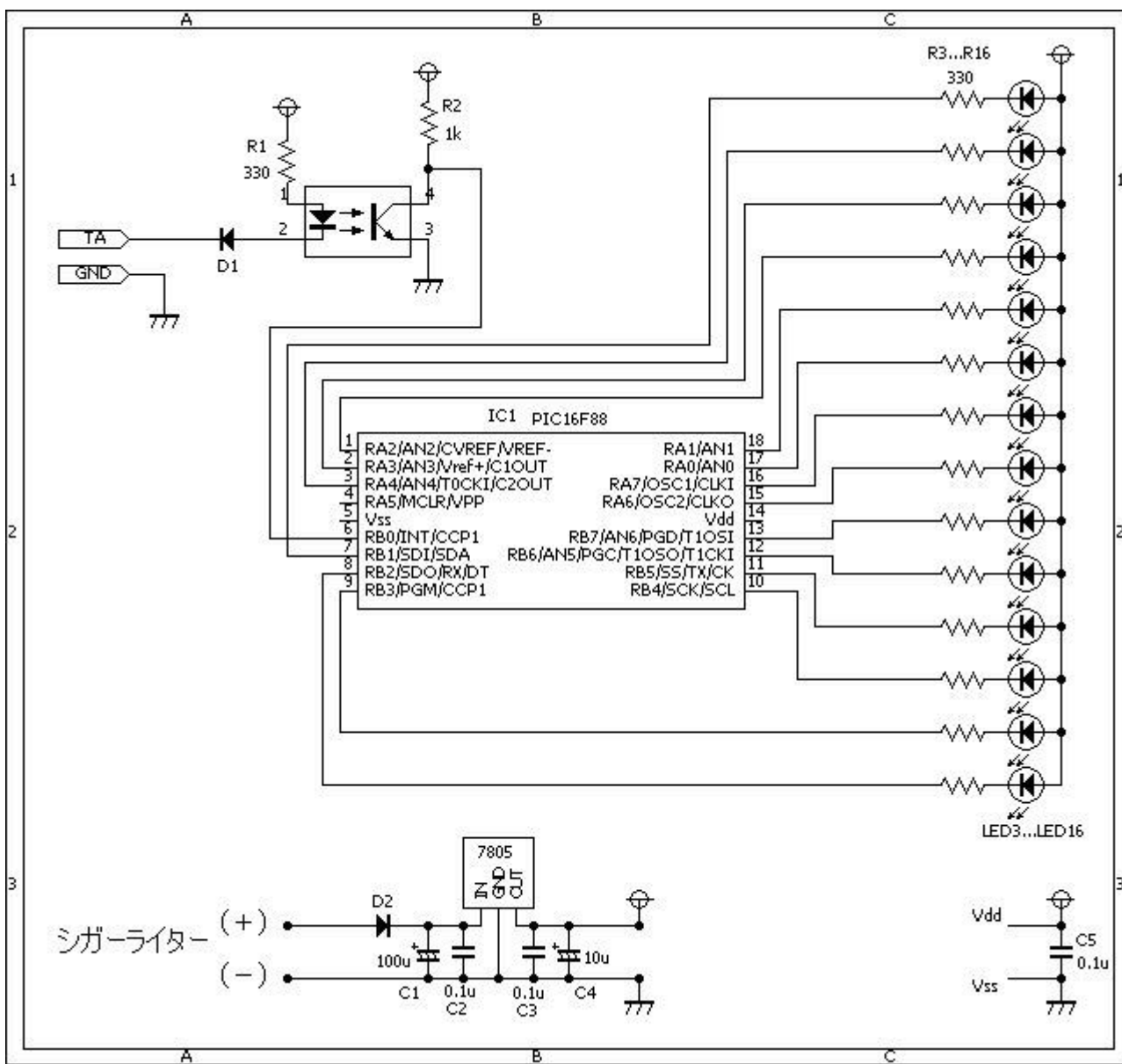
1. TA信号が、Lowになるまで待つ。
2. TA信号が、Highになるまで待つ。
3. TIMER1を開始させる。
4. TA信号が、Lowになるまで待つ。
5. TA信号が、Highになるまで待つ。

- (60秒 ÷ パルス幅) ÷ エンジン1回転あたりのパルス数

所を 適時変更してください。

```
#defineCYCLE DATA1.5
```

回路図



ソースコード

GP_TachoMeter.c

```
//*****
*
/*
『汎用タコメータ』
*/
//*****
*

#define LED01      PORTB.F2
#define LED02      PORTB.F3
#define LED03      PORTB.F4
#define LED04      PORTB.F5
#define LED05      PORTB.F6
#define LED06      PORTB.F7
#define LED07      PORTA.F6
#define LED08      PORTA.F7
#define LED09      PORTA.F0
#define LED10      PORTA.F1
#define LED11      PORTA.F2
#define LED12      PORTA.F3
#define LED13      PORTA.F4
#define LED14      PORTB.F1

#define ON          0
#define OFF         1

#define TA_SIG      PORTB.F0

#define CYCLE_DATA  1.5          //エンジン1回転あたりのパルス数

//*****
*

double measurement()
{
    static unsigned int cnt;
    // TIMER1の設定
    PIE1.TMR1IE = 0;
    PIR1.TMR1IF = 0;
    T1CON.T1RUN = 0;
    T1CON.T1CKPS0 = 1;
    T1CON.T1CKPS1 = 1;
    T1CON.T10SCEN = 0;
    T1CON.NOT_T1SYNC = 1;
    T1CON.TMR1CS = 0;
    T1CON.TMR1ON = 0;
    TMR1L = 0;
    TMR1H = 0;
    //信号の立ち上がりをチェックする。
```

```
while (TA_SIG != OFF)
;
while (TA_SIG != ON)
;
//TIMER1を開始する。
T1CON.TMR1ON = 1;
//信号の立ち上がりをチェックする。
while (TA_SIG != OFF)
;
while (TA_SIG != ON)
;
//TIMER1を停止する。
T1CON.TMR1ON = 0;
//オーバーフローをチェックする。
if (PIR1.TMR1IF == 1)
return (-1.0);
//実時間に変換する。
cnt = TMR1H << 8;
cnt = cnt | TMR1L;
return ((double)cnt * 4.0);    // 4.0usec=(1/8000000)*4*8*1000000
}

//*****
*

void BarDisp(int cnt)
{
static char i;
//
LED01 = LED02 = LED03 = LED04 = LED05 = LED06 = LED07 = LED08 =
LED09 = LED10 = LED11 = LED12 = LED13 = LED14 = OFF;
//
switch (cnt) {
case 14:
LED14 = ON;
case 13:
LED13 = ON;
case 12:
LED12 = ON;
case 11:
LED11 = ON;
case 10:
LED10 = ON;
case 9:
LED09 = ON;
case 8:
LED08 = ON;
case 7:
LED07 = ON;
case 6:
```

```

        LED06 = ON;
    case 5:
        LED05 = ON;
    case 4:
        LED04 = ON;
    case 3:
        LED03 = ON;
    case 2:
        LED02 = ON;
    case 1:
        LED01 = ON;
    case 0:
        break;
    case -1:
        for (i = 0; i < 5; i++) {
            LED01 = LED02 = LED03 = LED04 = LED05 = LED06 = LED07 = ON;
            Delay_ms(100);
            LED01 = LED02 = LED03 = LED04 = LED05 = LED06 = LED07 =
OFF;
            Delay_ms(100);
        }
        break;
    default:
        for (i = 0; i < 5; i++) {
            LED08 = LED09 = LED10 = LED11 = LED12 = LED13 = LED14 = ON;
            Delay_ms(100);
            LED08 = LED09 = LED10 = LED11 = LED12 = LED13 = LED14 =
OFF;
            Delay_ms(100);
        }
        break;
    }
}

//*****
*

void main()
{
    static char buf[6];
    static double cycleTime, minRotationCnt;
    //
    OSCCON = 0b01110000; // クロックは8Mhz
    CMCON = 0b00000111; // コンパレータは使用しない。
    ANSEL = 0b00000000; // □□□変換を使用しない。
    TRISA = 0b00100000;
    TRISB = 0b00000001;
    //
    while (1) {
        //パルス間隔の時間を測定する。
        cycleTime = measurement();
    }
}

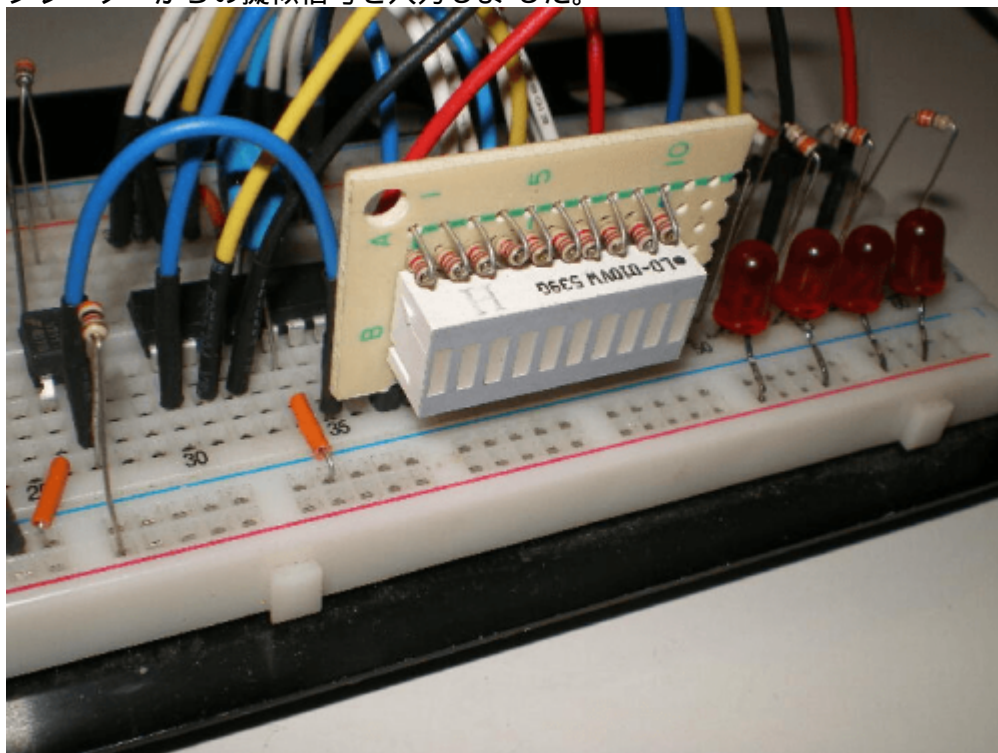
```

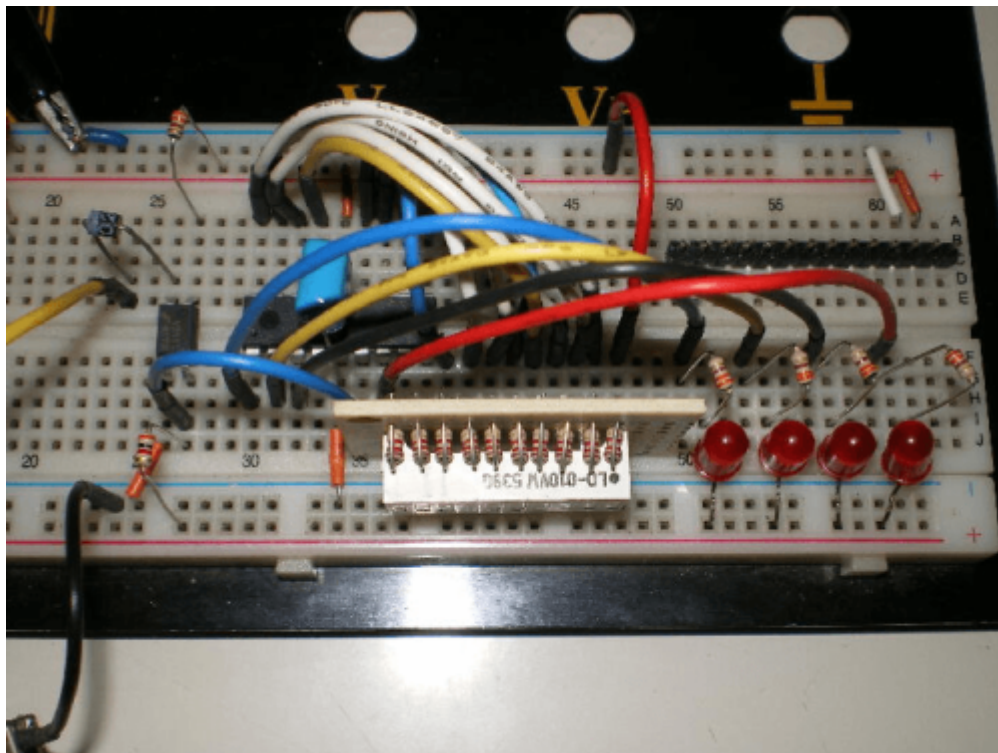
```
//オーバーフローであればエラー表示する。
if (cycleTime == -1.0) {
    BarDisp(-1);
    continue;
}
//1分間の回転数を求める。
minRotationCnt = (60000000.0 / cycleTime) / CYCLE_DATA;
//1000回転に換算し、バー表示する。
BarDisp((int)(minRotationCnt / 1000.0));
}
} //~!

//*****
*
```

動作確認

いつものブレッドボードで確認してみました。14個のLEDは、手持ちの関係で、10バーLEDアレイと普通の赤色LED4個としました。尚、確認に当たっては、実際の車のTA信号ではなく、手持ちのテストオシレーターからの擬似信号を入力しました。

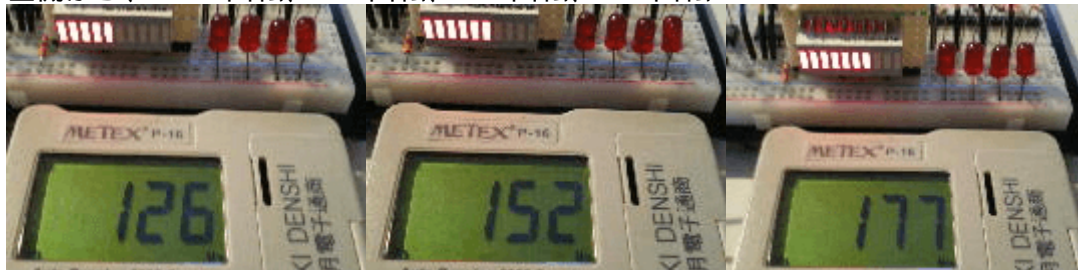


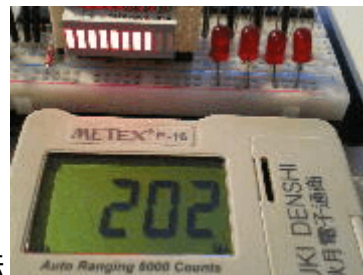


左側から、停止状態、1000回転、2000回転、3000回転(LEDの点灯と周波数の変化)



左側から、4000回転、5000回転、6000回転、7000回転

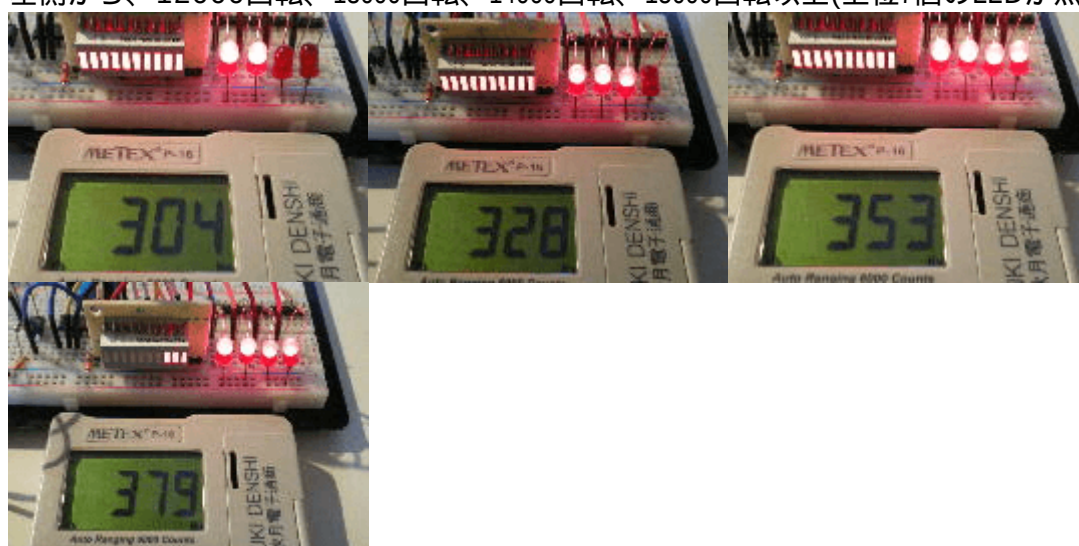




左側から、8000回転、9000回転、10000回転、11000回転



左側から、12000回転、13000回転、14000回転、15000回転以上(上位7個のLEDが点滅)



From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:105&rev=1588205921>

Last update: 2025/10/17 14:27

